

AMENDMENTS TO THE CLAIMS

This listing of claims will replace all prior versions, and listings, of claims in the application:

1. (previously presented) A method of dynamic installation and activation of software packages in a node in a distributed network of nodes, the method comprising the computer-implemented steps of:

storing, in a software package storage of a master node in the distributed network, a plurality of software packages and a plurality of software modules that the nodes in the distributed network will be using;

wherein each software package of the plurality of software packages contains at least one module and associated dependency information;

receiving a software update for a node on said master node;

wherein the software update contains a set of one or more software packages;

storing the software update on said software package storage;

wherein said master node notifies said node that a software update is being requested;

wherein said master node passes said node identities of one or more software packages to be updated and module dependencies;

wherein said node determines, using the module dependencies, running processes on said node that will be affected by the software update.
2. (previously presented) A method as recited in Claim 1, wherein each module has a binary signature.
3. (previously presented) A method as recited in Claim 1, wherein each node has a list of desired characteristics stored on said master node which is compared by said

master node to each module in the software update to determine which modules in the set of one or more software packages should be sent to a node.

4. (canceled).
5. (previously presented) A method as recited in Claim 1, wherein said node notifies affected processes that the software update is being requested; wherein each notified process evaluates the effect that the software update will have on its operation; wherein if any of the notified processes determine that the software update will degrade or have a negative impact on said node's normal operation, the process returns a veto to said node; and wherein if a process finds that the software update will have no negative effects, the process returns an acceptance of the software update to said node.
6. (previously presented) A method as recited in Claim 5, wherein said node waits for all of the notified processes to return results of their evaluations and once all of the processes have reported to said node, said node notifies said master node if any of the processes have vetoed the software update.
7. (previously presented) A method as recited in Claim 6, wherein if said master node receives an acceptance from said node then said master node sends the set of one or more software packages from said software package storage to said node.

8. (original) A method as recited in Claim 7, wherein said node immediately runs software package modules, by loading the modules from the software package(s) and signals processes that are being replaced by the modules and the affected processes that the changeover is going to occur; wherein when all of the signaled processes indicate that they are ready and waiting for the changeover, said node starts new modules and signals the affected processes that the changeover has occurred; wherein each module starts without affecting normal operation of said node; and wherein each affected process restarts, if required, without affecting normal operation of said node.
9. (previously presented) A method as recited in Claim 8, wherein said node continues with normal operations and notifies said master node that it has completed the software update; and wherein said master node checks the module dependencies to ensure that any inter-nodal and intra-node dependencies are complete.
10. (original) A method as recited in Claim 9, wherein if there are any discrepancies in the inter-nodal and intra-node dependencies, then said master node notifies a user.
11. (previously presented) A method as recited in Claim 8, further comprising storing, in the software package storage, older versions of the software packages and the software modules that are kept for regressing said node back to a previous module or software package version, wherein when said node does not store the set of one or more software packages in its local persistent storage, then said node can later regress back to previous modules stored in the local persistent storage if it restarts or said master node tells it to regress.

12. (previously presented) A method as recited in Claim 7, wherein said node extracts version information and dependency information from the set of one or more software packages and stores the information in its local persistent storage.
13. (previously presented) A method as recited in Claim 12, wherein said node compares binary signatures of modules in the set of one or more software packages with corresponding modules stored in the local persistent storage to discover which modules have been updated; wherein any binary signatures that match indicate that the module has not changed; and wherein any modules that have different binary signatures replace the corresponding modules stored in the local persistent storage.
14. (previously presented) A method as recited in Claim 13, wherein said node runs software package modules by loading the modules from the set of one or more software packages and signals processes that are being replaced by the modules and the affected processes that a changeover is going to occur; wherein when all of the signaled processes indicate that they are ready and waiting for the changeover, the node starts new modules and signals the affected processes that the changeover has occurred; wherein each module starts without affecting normal operation of said node; and wherein each affected process restarts, if required, without affecting normal operation of said node.
15. (previously presented) A method as recited in Claim 14, wherein said node continues with normal operations and notifies said master node that it has completed the

- software update; and wherein said master node checks dependencies of the set of one or more software packages to ensure that any inter-nodal and intra-node dependencies are complete.
16. (original) A method as recited in Claim 15, wherein if there are any discrepancies in the inter-nodal and intra-node dependencies, then said master node notifies a user.
17. (original) A method as recited in Claim 6, wherein if more than one node was being updated, the software update will not occur if any node vetoes the software update.
18. (previously presented) A method as recited in Claim 6, wherein if said master node receives a veto from said node, then said master node does not update said node and notifies a user that the software update will adversely affect said node.
19. (previously presented) A method as recited in Claim 18, further comprising receiving an indication to continue updating said node, wherein said master node forces said node to accept the software update.
20. (previously presented) A method as recited in Claim 1, further comprising initiating the software update by receiving an image containing the software update onto said master node.
21. (previously presented) A method as recited in Claim 20, receiving an indication of a set of nodes and a set of software packages that are to be updated.

22. (original) A method as recited in Claim 1, wherein the software update contains a list of nodes to be updated.
23. (original) A method as recited in Claim 1, wherein the software update contains a list of software packages destined for each node.
24. (previously presented) A method as recited in Claim 1, wherein the master node has an ability to categorize nodes into classes where all of the nodes in a particular class of nodes have a same software configuration and may have differing processor types.
25. (previously presented) A method as recited in Claim 1, wherein each software package of the plurality of software packages contains version information, dependency information, and other metadata information pertaining to software in the package.
26. (previously presented) A method as recited in Claim 25, wherein the other metadata information includes a list of application program interface (API) providers and consumers.
27. (previously presented) A computer-readable storage medium carrying one or more sequences of instructions for dynamic installation and activation of software packages in a node in a distributed network of nodes, which instructions, when executed by one or more processors, cause the one or more processors to carry out the steps of:

- storing, in a software package storage of a master node in the distributed network, a plurality of software packages and a plurality of software modules that the nodes in the distributed network will be using;
- wherein each software package of the plurality of software packages contains at least one module;
- receiving a software update for a node on said master node;
- wherein the software update contains a set of one or more software packages;
- storing the software update on said software package storage;
- wherein said master node notifies said node that a software update is being requested;
- wherein said master node passes said node identities of one or more software packages to be updated and module dependencies; and
- wherein said node determines, using the module dependencies, running processes on said node that will be affected by the software update.
28. (previously presented) A computer-readable storage medium as recited in Claim 27, wherein each module has a binary signature.
29. (previously presented) A computer-readable storage medium as recited in Claim 27, wherein each node has a list of desired characteristics stored on said master node which is compared by said master node to each module in the software update to determine which subset of modules should be sent to a node.
30. (canceled).

31. (previously presented) A computer-readable storage medium as recited in Claim 27, wherein said node notifies affected processes that the software update is being requested; wherein each notified process evaluates the effect that the software update will have on its operation; wherein if any of the notified processes determine that the software update will degrade or have a negative impact on said node's normal operation, the process returns a veto to said node; and wherein if a process finds that the software update will have no negative effects, the process returns an acceptance of the software update to said node.
32. (previously presented) A computer-readable storage medium as recited in Claim 31, wherein said node waits for all of the notified processes to return results of their evaluations and once all of the processes have reported to said node, said node notifies said master node if any of the processes have vetoed the software update.
33. (previously presented) A computer-readable storage medium as recited in Claim 32, wherein if said master node receives an acceptance from said node then said master node sends the set of one or more software packages from said software package storage to said node.
34. (previously presented) A computer-readable storage medium as recited in Claim 33, wherein said node immediately runs software package modules, by loading the modules from the software package(s) and signals processes that are being replaced by the modules and the affected processes that the changeover is going to occur; wherein when all of the signaled processes indicate that they are ready and waiting

for the changeover, the node starts new modules and signals the affected processes that the changeover has occurred; wherein each module starts without affecting normal operation of said node; and wherein each affected process restarts, if required, without affecting normal operation of said node.

35. (previously presented) A computer-readable storage medium as recited in Claim 34, wherein said node continues with normal operations and notifies said master node that it has completed the software update; and wherein said master node checks the module dependencies to ensure that any inter-nodal and intra-node dependencies are complete.
36. (previously presented) A computer-readable storage medium as recited in Claim 35, wherein if there are any discrepancies in the inter-nodal and intra-node dependencies, then said master node notifies a user.
37. (previously presented) A computer-readable storage medium as recited in Claim 34, wherein the sequences of instructions include instructions which, when executed by the one or more processors, further cause the one or more processors to perform storing, in the software package storage, older versions of a subset of the plurality of software packages and a subset of the plurality of software modules that are kept for regressing said node back to a previous software module or software package version, wherein when said node does not store the set of one or more software packages in its local persistent storage, then said node can later regress back to previous modules stored in the local persistent storage if it restarts or said master node tells it to regress.

38. (previously presented) A computer-readable storage medium as recited in Claim 33, wherein said node extracts version information and dependency information from the set of one or more software packages and stores the information in its local persistent storage.
39. (previously presented) A computer-readable storage medium as recited in Claim 38, wherein said node compares binary signatures of modules in the set of one or more software packages with corresponding modules stored in the local persistent storage to discover which modules have been updated; wherein any binary signatures that match indicate that the module has not changed; and wherein any modules that have different binary signatures replace the corresponding modules stored in the local persistent storage.
40. (previously presented) A computer-readable storage medium as recited in Claim 39, wherein said node runs software package modules by loading the modules from the set of one or more software packages and signals processes that are being replaced by the modules and the affected processes that a changeover is going to occur; wherein when all of the signaled processes indicate that they are ready and waiting for the changeover, the node starts new modules and signals the affected processes that the changeover has occurred; wherein each module starts without affecting normal operation of said node; and wherein each affected process restarts, if required, without affecting normal operation of said node.

41. (previously presented) A computer-readable storage medium as recited in Claim 40, wherein said node continues with normal operations and notifies said master node that it has completed the software update; and wherein said master node checks dependencies of the set of one or more software packages to ensure that any inter-nodal and intra-node dependencies are complete.
42. (previously presented) A computer-readable storage medium as recited in Claim 41, wherein if there are any discrepancies in the inter-nodal and intra-node dependencies, then said master node notifies a user.
43. (previously presented) A computer-readable storage medium as recited in Claim 32, wherein if more than one node was being updated, the software update will not occur if any node vetoes the software update.
44. (previously presented) A computer-readable storage medium as recited in Claim 32, wherein if said master node receives a veto from said node, then said master node does not update said node and notifies a user that the software update will adversely affect said node.
45. (previously presented) A computer-readable storage medium as recited in Claim 44, wherein the instructions include instructions which, when executed by the one or more processors, further cause the one or more processors perform receiving an indication to continue updating said node, wherein said master node forces said node to accept the software update.

46. (previously presented) A computer-readable storage medium as recited in Claim 27, wherein the one or more sequences of instructions are instructions which, when executed by the one or more processors, further cause the one or more processors to perform initiating the software update by receiving an image containing the software update onto said master node.
47. (previously presented) A computer-readable storage medium as recited in Claim 46, wherein the one or more sequences of instructions are instructions which, when executed by the one or more processors, further cause the one or more processors to perform the step of receiving an indication of a set of nodes and a set of software packages that are to be updated.
48. (previously presented) A computer-readable storage medium as recited in Claim 27, wherein the software update contains a list of nodes to be updated.
49. (previously presented) A computer-readable storage medium as recited in Claim 27, wherein the software update contains a list of software packages destined for each node.
50. (previously presented) A computer-readable storage medium as recited in Claim 27, wherein the master node has an ability to categorize nodes into classes where all of the nodes in a particular class of nodes have a same software configuration and may have differing processor types.

51. (previously presented) A computer-readable storage medium as recited in Claim 27, wherein each software package of the plurality of software packages contains version information, dependency information, and other metadata information pertaining to software in the package.
52. (previously presented) A computer-readable storage medium as recited in Claim 51, wherein the other metadata information includes a list of application program interface (API) providers and consumers
53. (previously presented) An apparatus, comprising:
a master node;
means for storing, in a software package storage of the master node, a plurality of software packages and a plurality of software modules that nodes in a distributed network will be using;
wherein each software package of the plurality of software packages contains at least one module;
means for receiving a software update for a node on said master node;
wherein the software update contains a set of one or more software packages;
means for storing the software update on said software package storage;
wherein said master node notifies said node that a software update is being requested;
wherein said master node passes said node identities of one or more software packages to be updated and module dependencies; and

wherein said node determines, using the module dependencies, running processes on
said node that will be affected by the software update.

54. (previously presented) An apparatus as recited in Claim 53, wherein each module has a binary signature.
55. (original) An apparatus as recited in Claim 53, wherein each node has a list of desired characteristics stored on said master node which is compared by said master node to each module in the software update to determine which subset of modules should be sent to a node.
56. (canceled).
57. (previously presented) An apparatus as recited in Claim 53, wherein said node notifies affected processes that the software update is being requested; wherein each notified process evaluates the effect that the software update will have on its operation; wherein if any of the processes determine that the software update will degrade or have a negative impact on said node's normal operation, the process returns a veto to said node; and wherein if a process finds that the software update will have no negative effects, the process returns an acceptance of the software update to said node.
58. (previously presented) An apparatus as recited in Claim 57, wherein said node waits for all of the notified processes to return results of their evaluations and once all of

the processes have reported to said node, said node notifies said master node if any of the processes have vetoed the software update.

59. (previously presented) An apparatus as recited in Claim 58, wherein if said master node receives an acceptance from said node then said master node sends the set of one or more software packages from said software package storage to said node.
60. (original) An apparatus as recited in Claim 59, wherein said node immediately runs software package modules, by loading the modules from the software package(s) and signals processes that are being replaced by the modules and the affected processes that the changeover is going to occur; wherein when all of the signaled processes indicate that they are ready and waiting for the changeover, the node starts new modules and signals the affected processes that the changeover has occurred; wherein each module starts without affecting normal operation of said node; and wherein each affected process restarts, if required, without affecting normal operation of said node.
61. (previously presented) An apparatus as recited in Claim 60, wherein said node continues with normal operations and notifies said master node that it has completed the software update; and wherein said master node checks the module dependencies to ensure that any inter-nodal and intra-node dependencies are complete.
62. (original) An apparatus as recited in Claim 61, wherein if there are any discrepancies in the inter-nodal and intra-node dependencies, then said master node notifies a user.

63. (currently amended) An apparatus as recited in Claim 60, further comprising means for storing, in the software package storage, older versions of a subset of the plurality of software packages and a subset of the plurality of the software modules that are kept for regressing said node back to a previous software module or software package version, wherein when said node does not store the ~~[[sset]]~~ set of one or more software packages in its local persistent storage, then said node can later regress back to previous modules stored in the local persistent storage if it restarts or said master node tells it to regress.
64. (previously presented) An apparatus as recited in Claim 59, wherein said node extracts version information and dependency information from the set of one or more software packages and stores the information in its local persistent storage.
65. (previously presented) An apparatus as recited in Claim 64, wherein said node compares binary signatures of modules in the set of one or more software packages with corresponding modules stored in the local persistent storage to discover which modules have been updated; wherein any binary signatures that match indicate that the module has not changed; and wherein any modules that have different binary signatures replace the corresponding modules stored in the local persistent storage.
66. (previously presented) An apparatus as recited in Claim 65, wherein said node runs software package modules by loading the modules from the set of one or more software packages and signals processes that are being replaced by the modules and the affected processes that a changeover is going to occur; wherein when all of the

- signaled processes indicate that they are ready and waiting for the changeover, the node starts new modules and signals the affected processes that the changeover has occurred; wherein each module starts without affecting normal operation of said node; and wherein each affected process restarts, if required, without affecting normal operation of said node.
67. (previously presented) An apparatus as recited in Claim 66, wherein said node continues with normal operations and notifies said master node that it has completed the software update; and wherein said master node checks dependencies of the set of one or more software packages to ensure that any inter-nodal and intra-node dependencies are complete.
68. (original) An apparatus as recited in Claim 67, wherein if there are any discrepancies in the inter-nodal and intra-node dependencies, then said master node notifies a user.
69. (original) An apparatus as recited in Claim 58, wherein if more than one node was being updated, the software update will not occur if any node vetoes the software update.
70. (previously presented) An apparatus as recited in Claim 58, wherein if said master node receives a veto from said node, then said master node does not update said node and notifies a user that the software update will adversely affect said node.

71. (previously presented) An apparatus as recited in Claim 70, further comprising means for receiving an indication to continue updating said node, wherein said master node forces said node to accept the software update.
72. (previously presented) An apparatus as recited in Claim 53, further comprising means for initiating the software update by receiving an image containing the software update onto said master node.
73. (previously presented) An apparatus as recited in Claim 72, further comprising means for receiving an indication of a set of nodes and a set of software packages that are to be updated.
74. (original) An apparatus as recited in Claim 53, wherein the software update contains a list of nodes to be updated.
75. (original) An apparatus as recited in Claim 53, wherein the software update contains a list of software packages destined for each node.
76. (previously presented) An apparatus as recited in Claim 53, wherein the master node has an ability to categorize nodes into classes where all of the nodes in a particular class of nodes have a same software configuration and may have differing processor types.

77. (previously presented) An apparatus as recited in Claim 53, wherein each software package of the plurality of software packages contains version information, dependency information, and other metadata information pertaining to software in the package.
78. (previously presented) An apparatus as recited in Claim 77, wherein the other metadata information includes a list of application program interface (API) providers and consumers.